[1 tool called]

[3 tools called]

Based on my analysis of all the controller functions, here's a comprehensive breakdown of the **types of unit tests** we should write:

**📋 Unit Test Types for All Controllers**

**🎯 userControllers.js (5 functions)**

**1. registerUser**

* ✅ **Core Functionality**: Database insertion via User.create()
* ✅ **Data Transformation**: ID generation, password hashing, timestamp setting
* ✅ **Error Handling**: Database failures, hashing failures

**2. updateUser**

* ✅ **Core Functionality**: Database update via User.findOneAndUpdate()
* ✅ **Data Transformation**: Filter creation, update object creation (removing user\_id)
* ✅ **Error Handling**: Database failures, user not found

**3. updateUserPassword**

* ✅ **Core Functionality**: Password update via User.findOneAndUpdate()
* ✅ **Data Transformation**: Password hashing with argon2
* ✅ **Error Handling**: Database failures, hashing failures

**4. getUserById**

* ✅ **Core Functionality**: Database query via User.findOne()
* ✅ **Data Retrieval**: Return user document or null
* ✅ **Error Handling**: Database failures

**5. searchUsersByName**

* ✅ **Core Functionality**: Database search via User.find()
* ✅ **Data Retrieval**: Return array of matching users
* ✅ **Error Handling**: Database failures

**🏠 shippingAddressControllers.js (5 functions)**

**1. createShippingAddress**

* ✅ **Core Functionality**: Subdocument creation via User.findOneAndUpdate() with $push
* ✅ **Data Transformation**: ID generation, subdocument validation
* ✅ **Error Handling**: User not found, empty request body, database failures
* ✅ **Validation**: User existence check, empty object check

**2. updateShippingAddress**

* ✅ **Core Functionality**: Subdocument update via User.updateOne() with positional operator
* ✅ **Data Transformation**: Query building, update object creation with $set
* ✅ **Error Handling**: User not found, address not found, empty request body
* ✅ **Validation**: Multiple existence checks

**3. getShippingAddressById**

* ✅ **Core Functionality**: Subdocument retrieval via User.aggregate()
* ✅ **Data Transformation**: Aggregation pipeline with $match, $unwind, $project
* ✅ **Error Handling**: User not found, address not found, empty request body
* ✅ **Response Handling**: Single result vs not found

**4. searchShippingAddress**

* ✅ **Core Functionality**: Subdocument search via User.aggregate()
* ✅ **Data Transformation**: Dynamic query building, aggregation pipeline
* ✅ **Error Handling**: User not found, no results found, empty request body
* ✅ **Response Handling**: Multiple results vs no results

**5. deleteShippingAddressById**

* ✅ **Core Functionality**: Subdocument deletion via User.updateOne() with $pull
* ✅ **Data Transformation**: Query building for deletion
* ✅ **Error Handling**: User not found, database failures
* ✅ **Response Handling**: Deletion result confirmation

**🌐 defaultController.js (1 function)**

**1. defaultController**

* ✅ **Core Functionality**: Simple response via res.send()
* ✅ **Response Handling**: Returns 'Hello World' string
* ✅ **Error Handling**: None (simple synchronous function)

**❌ urlNotFoundController.js (1 function)**

**1. urlNotFoundController**

* ✅ **Core Functionality**: Error creation and propagation
* ✅ **Error Handling**: Custom error creation with ResourceNotFoundError
* ✅ **Response Handling**: Calls next(error) to pass to global error handler
* ✅ **URL Handling**: Includes original URL in error message

**🎯 Test Categories Summary**

**Database Operations Tests**

// For each controller function

- Should call correct MongoDB operation (create, findOne, findOneAndUpdate, etc.)

- Should pass correct parameters to database

- Should handle database success

- Should handle database failures

**Data Transformation Tests**

// For functions that modify data

- Should generate IDs (createRandomString)

- Should hash passwords (argon2)

- Should create timestamps (getCurrentDateTime)

- Should build filters correctly

- Should create update objects correctly

- Should transform request data

**Error Handling Tests**

// For all async functions

- Should handle database connection failures

- Should handle validation errors

- Should handle resource not found errors

- Should handle empty request body errors

- Should propagate errors correctly

**Response Handling Tests**

// For all controller functions

- Should set correct HTTP status codes

- Should return correct response format

- Should handle null/empty results

- Should handle error responses

**Validation Tests**

// For functions with validation logic

- Should validate user existence

- Should validate resource existence

- Should validate request body content

- Should handle missing parameters

**📊 Test Priority by Function**

**Priority 1 (Critical - Must Test):**

1. ✅ registerUser - User creation (security critical)
2. ✅ updateUser - User updates (data integrity)
3. ✅ updateUserPassword - Password security (critical)
4. ✅ createShippingAddress - Address creation (business logic)

**Priority 2 (Important - Should Test):**

1. ✅ getUserById - User retrieval
2. ✅ searchUsersByName - User search
3. ✅ updateShippingAddress - Address updates
4. ✅ getShippingAddressById - Address retrieval

**Priority 3 (Nice to Have):**

1. ✅ searchShippingAddress - Address search
2. ✅ deleteShippingAddressById - Address deletion
3. ✅ defaultController - Simple response
4. ✅ urlNotFoundController - Error handling

**🔧 Common Test Patterns**

**For Database Operations:**

// Mock the database operation

User.create.mockResolvedValue(mockResult);

// Call the function

await controllerFunction(req, res, next);

// Verify database was called

expect(User.create).toHaveBeenCalledWith(expectedData);

**For Error Handling:**

// Mock database failure

User.create.mockRejectedValue(new Error('DB Error'));

// Call the function

await controllerFunction(req, res, next);

// Verify error is handled

expect(next).toHaveBeenCalledWith(expect.any(Error));

**For Response Handling:**

// Call the function

await controllerFunction(req, res, next);

// Verify response

expect(res.status).toHaveBeenCalledWith(201);

expect(res.json).toHaveBeenCalledWith(expectedData);

**📈 Estimated Test Count**

| **Controller** | **Functions** | **Tests per Function** | **Total Tests** |
| --- | --- | --- | --- |
| userControllers | 5 | ~8 each | ~40 tests |
| shippingAddressControllers | 5 | ~10 each | ~50 tests |
| defaultController | 1 | ~3 | ~3 tests |
| urlNotFoundController | 1 | ~5 | ~5 tests |
| **TOTAL** | **12** |  | **~98 tests** |

This comprehensive test suite would provide **complete coverage** of all controller functionality! 🎯